НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютеризовані системи управління»
спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»
на тему: «Система автоматизації освітнього процесу у школі»

Виконав:
студент IV курсу, групи ІА-61
Чижик Віктор Володимирович _____

Керівник:
асистент
Дорога-Іванюк Олена Олександрівна _____

Рецензент:
доцент кафедри ПЗКС ФПМ, к.т.н.
Цуркан Василь Васильович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»

Освітньо-професійна програма «Комп'ютеризовані системи управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Ролік
«___»_____2020 р.

ЗАВДАННЯ
на дипломний проєкт студенту
Чижику Віктору Володимировичу

1.Тема проєкту «Система автоматизації освітнього процесу у школі», керівник проєкту асистент Дорога Олена Олександрівна, затверджені наказом по університету від «_07_» _травня_2020 р. № _1081-с____

2. Термін подання студентом проєкту _09 червня 2020 року_____

3. Вихідні дані до проєкту

Система керування версіями, мови програмування Scala та JavaScript, бібліотеки JavaScript jQuery, середовище програмування Intellij IDEA, обраний фреймворк для розробки – Akka, СУБД – PostgreSQL.

4. Зміст пояснювальної записки

Опис предметної області, аналіз існуючих рішень, формування вимог до програмного забезпечення, вибір технологій розробки, розробка застосунку, тестування програмного забезпечення, документація для користувача, впровадження та використання розробленої системи.

5. Перелік графічного матеріалу

Діаграма використань, діаграма компонентів, діаграма послідовності, діаграма структури бази даних

6. Дата видачі завдання 02 лютого 2020 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вибір тематичного напрямку та узгодження теми дипломного проєкту	27.02.2020	
2	Аналіз теоретичних матеріалів та вивчення предметної області	13.04.2020	
3	Аналіз існуючих рішень	20.04.2020	
4	Аналіз вимог до програмного забезпечення	27.04.2020	
5	Вибір технологій розробки	03.05.2020	
6	Розробка автоматизованої системи	10.05.2020	
7	Налагодження та перевірка програми	17.05.2020	
8	Оформлення пояснювальної записки	20.06.2020	
9	Передзахист дипломного проєкту	25.05.2020	
10	Доопрацювання пояснювальної записки та підготовка презентації	30.05.2020	
11	Захист дипломного проєкту	09.06.2020	

Студент Віктор ЧИЖИК

Керівник проєкту Олена ДОРОГА-ІВАНЮК

АНОТАЦІЯ

Чижик В.В. Система автоматизації освітнього процесу у школі. КПІ ім. Ігоря Сікорського, Київ, 2020.

Проєкт містить 60 с. тексту, 15 рисунків, 21 таблицю, посилання на 20 літературних джерел, додатки та 4 конструкторських документів.

Ключові слова: Akka, Scala, Git, система керування версіями, JavaScript, діаграма використань.

Об’єктом розробки є система автоматизації освітнього процесу у школі.

Мета розробки – впровадження автоматизації в освітній процес.

У дипломному проєкті проведено ретельний аналіз та вибір технологій розробки системи, які надають можливість швидкої та якісної розробки програмного забезпечення. Розроблено систему автоматизації освітнього процесу у школі з використанням фреймворку Akka, мов програмування Scala та JavaScript.

Отримані результати можуть бути корисними при створенні аналогічних чи подібних систем.

SUMMARY

Chyzhyk V.V. Automation system of educational process in school. Igor Sikorsky KPI, Kyiv, 2020.

The project contains 60 pages. text, 15 figures, 21 tables, references to 20 literature sources, appendices and 4 design documents.

Keywords: Akka, Scala, Git, version control system, JavaScript, use-case diagram.

The object of development is a automation system of educational process in school.

The purpose of the development - introduction of automation in the educational process.

The graduation project has a thorough analysis and a choice of technologies of development system which give the chance of fast and high-quality software development is carried out. A system of automation of the school educational process has been developed using the Akka framework, Scala and JavaScript programming languages.

The obtained results can be useful in creation of analogous or similar systems.

Пояснювальна записка
до дипломного проєкту
на тему: «Система автоматизації освітнього процесу у школі»

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

AJAX – (англ. Asynchronous Javascript and XML) асинхронный JavaScript и XML

API – (англ. Application Programming Interface) програмний інтерфейс додатку

CI – (англ. Continuous Integration) неперервна інтеграція

DOM – (англ. Document Object Model) об’єктна модель документа

HTTP – (англ. HyperText Transfer Protocol) протокол передачі гіпертексту

HTTPS – (англ. HyperText Transfer Protocol Secure) – захищений протокол передачі тексту

СУБД – система управління базою даних

DI – (англ. Dependency Injection) – впровадження залежності

ВСТУП

З плином часу людство старається покращити своє життя за допомогою різноманітних інновацій. Доступ до мережі інтернет є у кожному місці на планеті. З кожним днем зростає кількість нових приладів, програм та методологій які позитивно впливають на розвиток як інформаційний технологій так і людського життя.

Великих обертів і масове застосування інформаційні технології набирають не тільки для спрощення або оптимізації потреб, а й під час глобальних проблем таких як пандемія. Головною метою проєкту є автоматизація освітнього процесу, адже в умовах карантину та інформаційної ери дане питання є надзвичайно актуальним.

Ефективне управління освітнім процесом є одним з найважливіших завдань у кожному навчальному закладі. Воно охоплює велику кількість осіб, залучених до цього процесу – учнів, викладачів, навчально-допоміжного та адміністративно-управлінського персоналу. Впливає на умови їх праці та навчання. В звичайних школах на даний час ведуть паперові звіти та документацію, оцінки учнів, відвідування, важливі повідомлення, звернення та домашнє завдання. Усе це можна автоматизувати за допомогою інформаційних технологій. Варто відмітити, що навчально освітній процес можна розглядати у вигляді «live» курсів. На сьогодні існує багато технологій дистанційного навчання у яких є широкий спектр функцій та інструментів для часткової автоматизації, але навчально освітній процес проводиться в денному форматі, що означає відсутність аналогів автоматизованих освітніх систем.

Особистою метою даного проєкту є поглиблення знань в новітніх, швидких та зростаючих технологіях, а також можливість створення універсального програмного забезпечення задля покращення освітньої програми навчання та усунення паперових процесів за допомогою автоматизованих освітніх систем що переходять від програмного забезпечення до хмарних обчислень. Також метою проєкту є створення абстрактного застосунку, що дасть змогу розширювати

програмне забезпечення методом добавлення нових компонентів, а також розробка в форматі мікросервісної архітектури.

Завданням цієї роботи є створення автоматизованої системи керування та документації освітнього процесу навчання з зручним інтерфейсом, швидкою роботою і мінімальною ресурсно вимогливою, яку можна використовувати в будь якому навчальному закладі. Також основною вимогою є підтримка даного застосунку на різноманітних платформах, що дає змогу використовувати вже існуючі програмні рішення які будуть виступати в ролі клієнта для користувачів. Важливим аспектом до проєкту є дотримання фаз життєвого циклу створення програмного забезпечення: дослідження предметної області, опрацювання вимог, проектування, реалізація з використанням передових технологій, тестування та експлуатація, що повинно значно підвищити відмовостійкість, збільшити безпеку передачі даних, забезпечити швидке розгортання застосунку та зменшити можливі відхилення.

Бакалаврський проєкт містить наступні розділи: вступ, базові розділи, висновки, список використаних джерел із 20 найменувань та 1 додатку. Графічна частина складається з 4 креслеників формату А3. Загальний обсяг – 60 сторінок.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Обґрунтування доцільності розробки

Системи автоматизації освітнього процесу у школі є важливим аспектом в підвищенні якості навчання. Впровадження таких систем економить час ведення документації та звітів, надає можливість швидкого аналізу успішності за будь який період часу, економить кошти, адже скільки ресурсів витрачається на рік задля забезпечення інструментами ведення освітнього процесу.

Існують варіації реалізації систем які допомагають автоматизувати освітній процес, але їхня основна ціль полягає в віддаленому навчанні. Проблема в тому що звичайні школи не можуть перейти на такий формат, адже це складно і дорого. Живе спілкування це дуже важливий аспект розвитку і на віддаленій основі без обґрунтованої потреби навчання буде недоцільним, тому дана розробка спрямована на часткову оптимізацію – початок впровадження інформаційних технологій в шкільному освітньому процесі.

Наступним кроком після часткової автоматизації є розширення функціональності вчителів, поширення навчального контенту в онлайн форматі, переведення книг в електронний формат, імплементація відео-звернень, можливість проведення зборів для батьків в онлайн форматі, можливість створення різноманітних курсів.

1.1.1 Опис та аналіз предметного середовища

Системи автоматизації освітнього шкільного процесу - це програмне забезпечення головною функцією якого є надання можливості ведення документації та перегляд історії за тривалий час. За допомогою історії можна проводити аналіз та вести певну статистику.

Системи даного роду будуються на основі клієнт-серверної архітектури. Система повинна бути розподіленою та масштабованою, що дає змогу проводити

маніпуляції над великим потоком даних. Всі розподілені мікро-сервіси повинні бути зв’язані між собою та реплікувати дані, щоб система була відмовостійкою.

Вона складається з таких компонентів:

— сервери, що виступають в ролі сховища даних та їхнього життєвого циклу;

— клієнти, які запитують дані, та постачають їх в залежності від прав доступу;

— мережа, надає можливість взаємодії між клієнтами та серверами, а також якщо система широко маштабована надає взаємодію між мікро-сервісами.

Використання автоматизації освітнього процесу у школі варто розглянути з двох сторін: зі сторони розробника та зі сторони користувача системи.

Перевагою розробника є те що в минулому він здобув шкільну освіту і аналіз та дослідження предметної області проведено в шкільні роки. Також дане питання не обмежує в технологіях імплементації, а програмне рішення є простим через те що основна ціль – це забезпечення доставки/збору даних. Написання абстракцій спрощує реалізацію системи і впровадження нового функціоналу є швидким. Також можливість інтегрування різних існуючих сервісів, таких як Telegram спрощує задачу написання взаємодії з клієнтом.

Перевагою користувача є зручний інтерфейс, можливість завантажувати дані в різних структурованих форматах. Також, якщо є інтеграції з сторонніми додатками, які вже використовуються ним, надають можливість «в один клік» почати використовувати дане програмне забезпечення.

1.1.2 Аналіз функціональних особливостей системи

Система автоматизації освітнього процесу у школі може являти собою web-сайт, desktop програму або телефонний додаток з обширною функціональністю.

Складається дана система з розділів які можна співвіднести з привілегіями користувачів:

— адміністративна панель;

— панель вчителів;

— панель учнів;

— панель батьків та інших користувачів.

Кожен з розділів надає користувачеві певного типу різноманітну функціональність яка залежить від його прав та можливостей.

Адміністративна панель дає змогу розширювати кількість користувачів та збільшувати діапазони даних якими можуть оперувати інші.

Панель вчителів надає можливість оцінювати учня та ділитися інформацією для нього.

Панель учнів слугує для отримання персональної інформації та успішності за певний період часу, а також загальнодоступної інформації для всіх учнів.

Панель батьків призначена для інформування про успішність дітей, а також для передачі важливих організаційних повідомлень.

До функціональних особливостей системи автоматизації освітнього шкільного процесу відноситься така функціональність, яку потрібно реалізувати:

— рівні доступу в залежності від типу користувачів;

— навігаційна панель;

— система ідентифікації та реєстрації;

— панелі для кожного з рівнів доступу;

— можливість інтеграції сторонніх ресурсів.

1.2 Призначення та задачі розробки

Розроблений продукт призначений для великої кількості шкільних навчальних закладів та може бути використаний як основа для повної автоматизації шкільного освітнього процесу. Також може бути використаний для персональних вчителів, для побудови процесу навчання, після певних модифікацій може бути використаний для вищих навчальних закладів.

Головна ціль цього застосунку це автоматизація процесів які в час цифрового століття є застарілими. Основною задачею є створення зручного, ресурсно економного програмного забезпечення з можливістю додавання нових модулів для майбутньої модифікації, яке можливо стане початком до повної автоматизації

шкільних процесів. Також основним аспектом є поділ користувачів на групи по рівнях доступу. У кожного рівня відповідно імплементація власного набору інструментів, швидке впровадження його в освітній процес у школі, впровадження без особливих навиків розробника.

1.4 Висновки до розділу

По-перше, за останній період часу інтернет та програмне забезпечення значно автоматизувало усі процеси. На жаль, існує багато осередків які не повністю або меншою мірою використовують плюси XXI століття.

По-друге, впровадження програмного забезпечення та автоматизація в цілому є дорогий та ресурсно затратний процес. Тому, кращим рішенням буде проведення часткових робіт в цьому напрямку.

По-третє, автоматизація шкільного освітнього процесу не тільки спрощує та оптимізовує процеси ведення документації, заповнення звітів, надсилання домашнього завдання медіа формату, надсилання звернень до персоналу навчального закладу, а й надає надійний захист даних та кращу структуризацію, а саме створення копій та зберігання на різних серверах.

Можна відзначити, що людина на сьогоднішній день безцінний ресурс, тому варто підкреслити що основним пріоритетом є саме автоматизація задля збільшення якості та часу на роботу в конкретній області діяльності.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

2.1 Існуючі системи автоматизації освітнього процесу

Далі будуть розглянути можливі варіанти популярних систем автоматизації освітнього процесу, а саме:

— Дистанційна система навчання «Moodle»;

— Дистанційна система навчання «Edmodo»;

— Дистанційна система навчання «Google Classroom»;

— Дистанційна система навчання «iSpring Online»;

— Дистанційна система навчання «OnLineTestPad»;

— Дистанційна система навчання «Quick Skills».

Варто відмітити що на сьогодні глобальних систем автоматизації освітнього процесу у школі немає. Тобто усі вони локальні і розроблені під певні запити кожного навчального закладу. Тому нижче розглянено відомі дистанційні системи навчання.

Усі дистанційні системи частково автоматизують освітній процес у школі, але вони не повною мірою описують денний формат навчання.

2.1.1 Дистанційна система навчання «Moodle»

Дистанційна система навчання «Moodle» – система яка створена для дистанційного навчання. Moodle є безкоштовною системою з відкритим кодом. Це дає змогу розробникам завантажувати, змінювати, створювати додаткові модулі та налаштовувати дане програмне забезпечення під власні потреби. Головним аспектом використання є широкий спектр можливостей та його масштабованість. Спільнота розробників впроваджує нові доповнення, які також є безкоштовними. Для початку використання доповнень достатньо завантажити та встановити для своєї системи. Прикладом доповнень є:

— модулі різноманітних конференцій;

— аудіо та відео чати;

— масова надсилання повідомлень;

— засоби проектної роботи;

— електронне портфоліо.

На рисунку 2.1 зображено головну сторінку користувача в дистанційній системі Moodle.

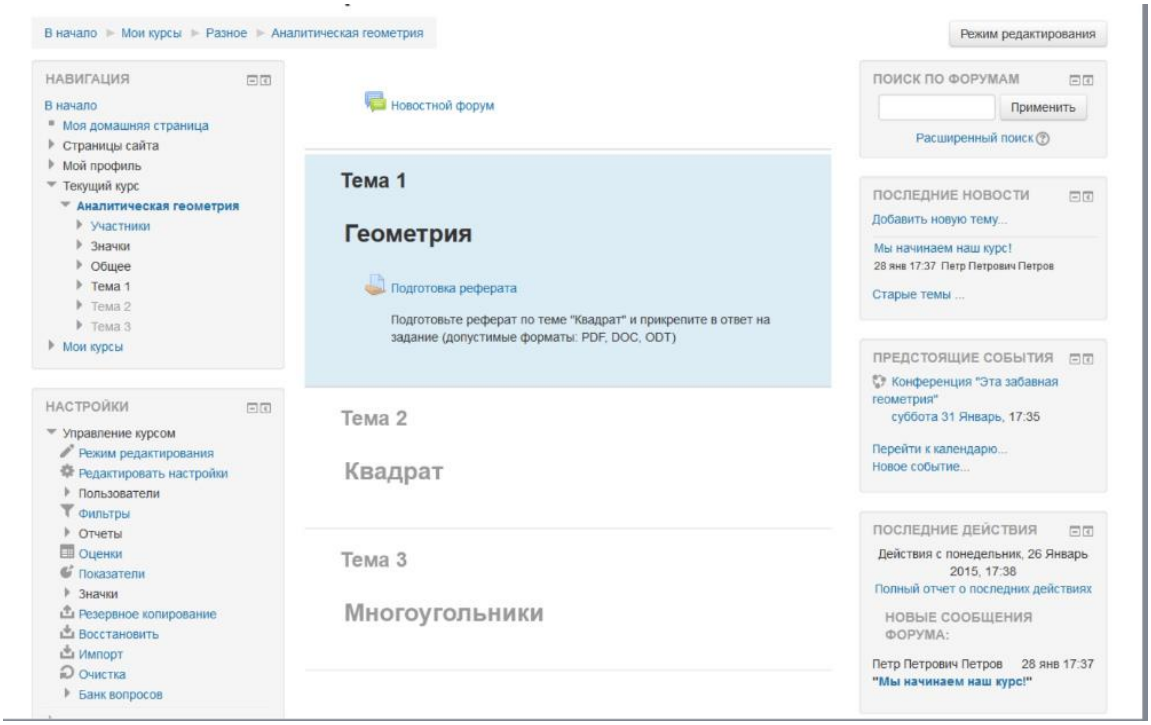

Рисунок 2.1 – Головна сторінка користувача в дистанційній системі Moodle

До переваг Moodle можна віднести:

— безкоштовна система, готова до використання;

— створення ресурсів для дистанційного навчання;

— широкі можливості керуванням курсами;

— містить потужний апарат тестування;

— включає в собі різновидність учбових елементів;

— дозволяє реалізувати диференційоване навчання;

— підтримує різновид педагогічних сценаріїв;

— підтримує налаштування керуванням доступу;

— візуальне відображення прогресу учня;

— можливість публікації різного виду інформації;

— wiki – інструмент для спільної роботи;

— велика кількість тестових завдань;

— лекції з технологією зворотного зв’язку.

Основні недоліки:

— система безкоштовна, але для її встановлення потрібен сервер;

— великі потреби до потужності сервера;

— потребує велику кількість ресурсів;

— не потрібний інструментарій;

— потребує навичок для встановлення та налаштування.

2.1.2 Дистанційна система навчання «Edmodo»

Програмний продукт Edmodo є web сайтом, який слугує для проведення уроків та лекцій. Учитель створює групу, кожна група має свій унікальний код та посилання які потрібно повідомити іншим учасникам освітнього процесу. Група має список учбових елементів: записи в вигляді тексту або файлів, тести, завдання та опитування. Можливість імпортування інформації з інших онлайн ресурсів: YouTube, новини з шкільного сайту, та інформацію з сторонніх сервісів. На рисунку 2.2 зображено сторінку результатів тестування користувача системи Edmodo.

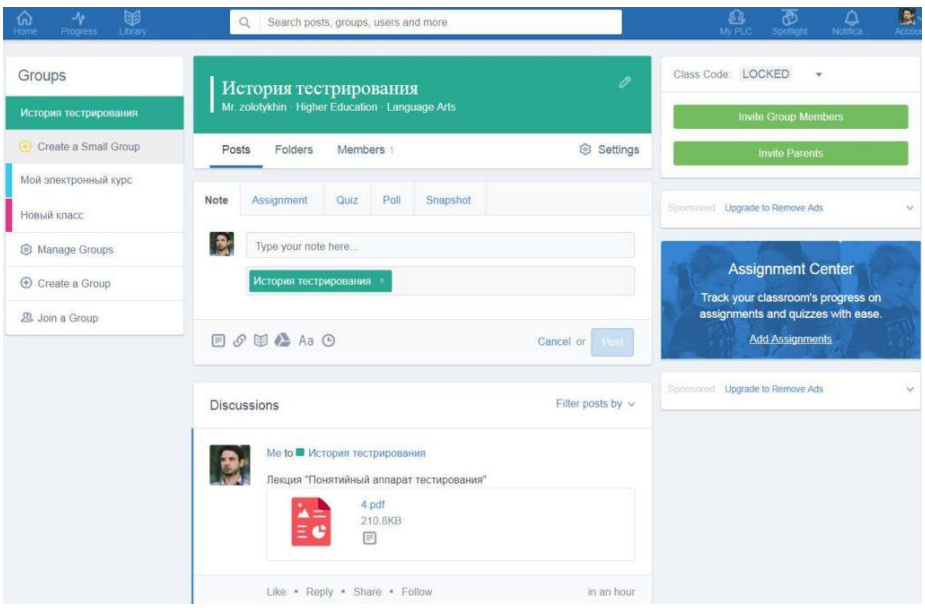

Рисунок 2.2 – Сторінка результатів тестування користувача системи Edmodo

З переваг Edmodo варто відзначити:

— безкоштовний;

— не містить реклами;

— проста реєстрація;

— створення курсів

— збереження ресурсів онлайн

— наявність відео-конференцій

— перевірка успішності учня

— користувачі діляться на 3 групи: вчителі, учні та батьки.

Недоліки:

— відсутність української та російської мови для налаштування;

— групи не можна об’єднувати, що сприяє великій кількості ключів та посилань;

— малий інструментарій учбових елементів.

2.1.3 Дистанційна система навчання «Google Classroom»

Дистанційна система навчання «Google Classroom» – продукт одного з лідерів IT індустрії. Варто відмітити що Google раніше мав великий набір інструментів пов’язаних з освітою та навчанням. Їхнє об’єднання стало етапом створення Google Classroom. Організація та налаштування потребує великих зусиль та спеціальних навичок. Немає можливості гнучкого налаштування, що призводить до перебудови освітнього процесу. Також варто відмітити особливості системи Google Classroom:

— використання Google інструментів (Google диск, Google Doc …);

— створення власного класу або курсу

— поширення навчальних матеріалів

— поширення пропозицій

— організація спілкування між учасниками процесу

— у користувачів освітнього процесу на Google диску створюється спільна папка під назвою «Клас»;

— папка «Клас» створюється для будь якої кількості студентів.

На рисунку 2.3 зображено основну сторінку користувача Google Classroom.

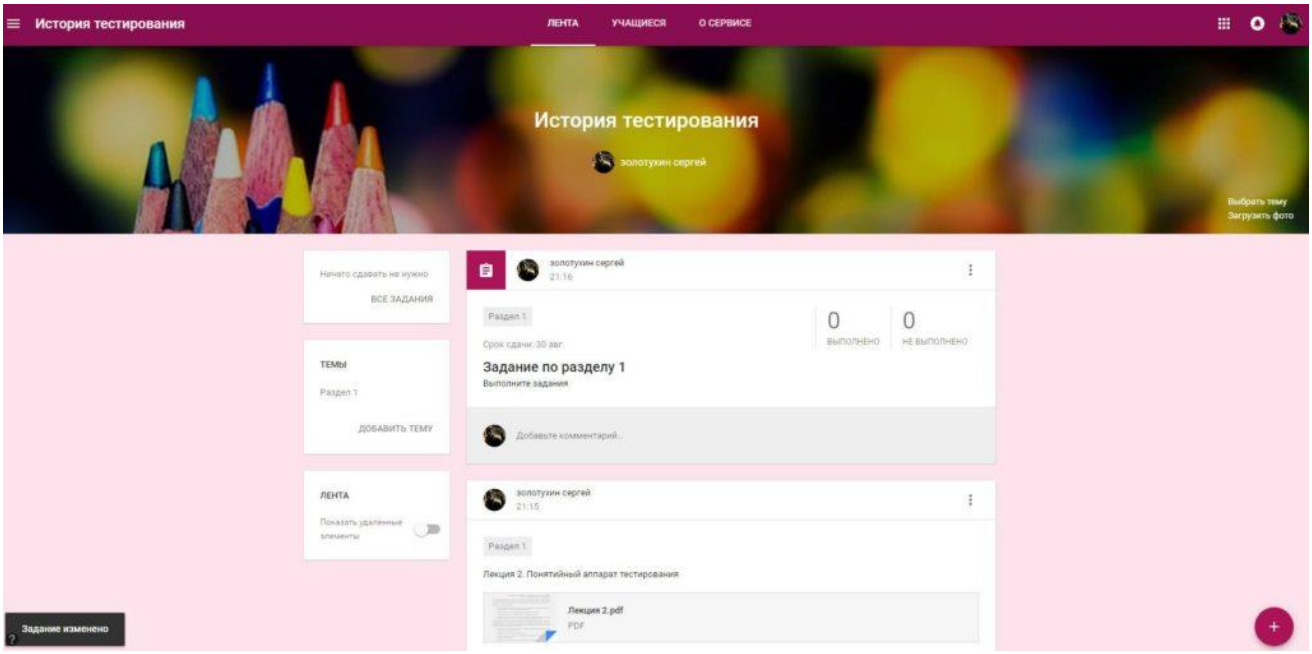

Рисунок 2.3 – Основна сторінка користувача Google Classroom

Переваги дистанційної системи навчання Google Classroom:

— підтримка української мови;

— безкоштовний;

— бренд;

— створювався для шкіл, Moodle в свою чергу підходить для вузів;

— реалізовані традиційні функції;

— можливість опублікувати учбовий матеріал, виставляти оцінки в журналі.

Недоліки:

— малий арсенал учбових інструментів;

— важко структуровані посилання;

— недостатня функціональність інтерфейсу.

2.1.4 Дистанційна система навчання «iSpring Online»

iSpring Online – система для організації дистанційного навчання. Дозволяє швидко реалізувати принципи дистанційного навчання в навчальному закладі. Надає можливість реєстрації, зберігання та збір інформації в режимі online. Не потребує ресурсів для встановлення – працює в online форматі. На рисунку 2.4 зображено сторінку з курсами користувача.

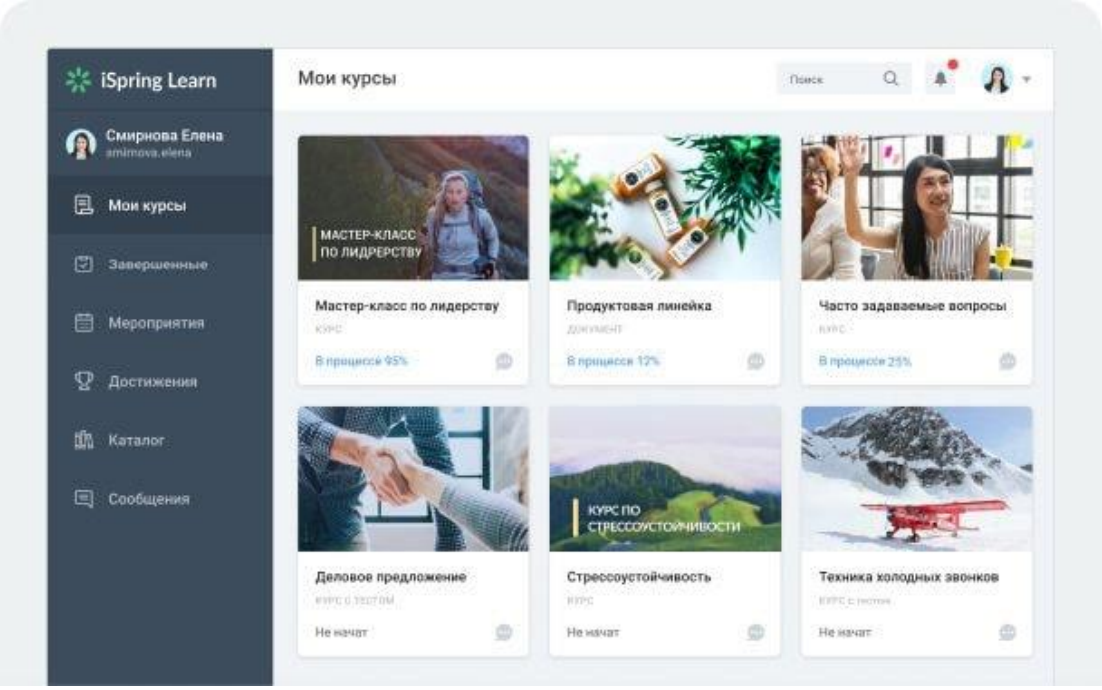

Рисунок 2.4 – Система дистанційного навчання iSpring Online

Переваги:

— безмежне хмарне сховище;

— інтеграція на більшість девайсів;

— контроль успішності;

— підтримка;

— впровадження власного дизайну;

— можливість вести лекції, демонструвати презентації.

Недоліки:

— платна система;

— мінімальна кількість користувачів – 50.

2.1.5 Дистанційна система навчання «OnLineTestPad»

Дистанційна система навчання «OnLineTestPad» – це безкоштовний сервіс для проведення тестів в online форматі. Є можливість збереження даних з всіх спроб користувачів, оцінок, вірних та невірних відповідей.

Переваги OnLineTestPad:

— велика кількість тестових завдань;

— гнучкі налаштування;

— наявність різноманітних обмежень.

Недоліки:

— велика кількість реклами;

— немає додаткових модулів для навчання.

2.1.6 Дистанційна система навчання «Quick Skills»

Дистанційна система навчання «Quick Skills» – це платний сервіс для проведення курсів в online форматі. Є можливість створення курсів в ігровому форматі. На рисунку 2.5 зображено основу сторінку сайту дистанційної системи навчання Quick Skills.

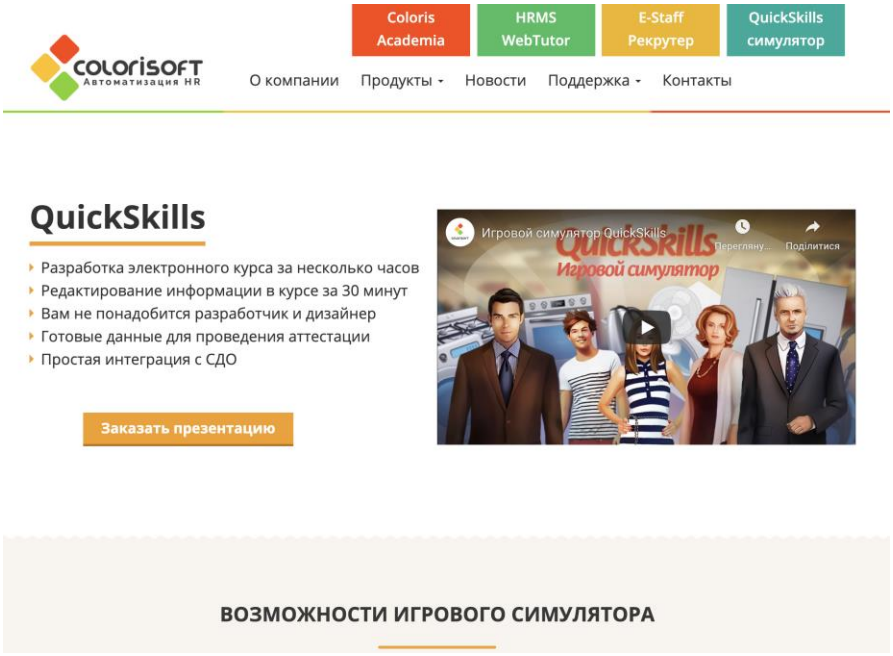

Рисунок 2.5 – Основа сторінку сайту Quick Skills

Переваги Quick Skills:

— швидке створення онлайн курсів;

— створення занять в режимі гри;

— наявність ігрових балів та рейтингів;

— готові дані для атестації;

— розбиття класу по групам;

— створення ресурсів для окремих рівнів доступу;

— широкі можливості керуванням курсами;

— впровадження власного дизайну.

Недоліки:

— платна система

— реклама

— малий набір інструментів для користувачів

— відстутність інтеграції з іншими програмними застосунками

— відсутність публікації та збереження даних

— відсутність перевірки успішності

— система в процесі розроблення, що сприяє виникненні випадкових похибок

2.2 Аналіз і порівняння

Судячи з розглянутих вище застосунків можна зробити висновок що існує багато варіацій систем автоматизації освітнього процесу. У кожного з них є свої переваги та недоліки, також варто відмітити що усі вище розглянуті рішення працюють в онлайн форматі. Після їхнього аналізу можна виділити основну функціональність яка повинна бути у розробленій системі, а саме:

— зрозумілий інтерфейс;

— швидкість роботи;

— простота налаштування та запуску;

— наявність різних типів доступу;

— створення окремих груп користувачів;

— створення курсів;

— інтеграція на різні середовища;

— можливість реалізації додаткових модулів;

— високий ступінь безпеки;

— широка функціональність.

2.3 Висновки до розділу

Для створення власного продукту, проведено огляд та аналіз існуючих автоматизованих системи освітнього процесу, відтворено позитивні аспекти кожного застосунку.

Також взято до уваги відгуки користувачів уже існуючих програмних продуктів, щоб на стадії проектування уникнути помилок та збільшити функціональність.

Усе вище розглянуте програмне забезпечення було спрямоване більше на дистанційне навчання, але шкільний освітній процес потребує денної форми. Тому умови проведення є також важливим аспектом.

3 РОЗРОБКА ВИМОГ ДО СТВОРЮВАНОЇ СИСТЕМИ

3.1 Розробка функціональних вимог до системи

Функціональні вимоги до системи є відповідними щодо варіантів використання:

— авторизація адміністратора;

— вихід адміністратора з системи;

— авторизація звичайних користувачів за допомогою Telegram;

— маніпуляції адміністратором над іншими користувачами;

— створення нових адміністраторів;

— маніпуляції адміністратором над шкільним семестром;

— маніпуляції адміністратором над навчальним планом;

— маніпуляції адміністратором над шкільними предметами;

— імпортування даних для швидкого створення плану;

— маніпуляції вчителем над домашнім завданням;

— створення вчителем успішності учнів;

— маніпуляції адміністратором над успішністю учнів;

— додавання нових оцінок та відвідувань вчителем;

— створення адміністратором та вчителем повідомлень;

— відображення даних для учня: успішність, домашнє завдання, повідомлення, розклад занять;

— відображення розкладу занять для вчителя;

— відображення успішності власних учнів для вчителя;

— відображення звернень для батьків;

— відображення успішності дітей для батьків.

3.2 Розробка нефункціональних вимог до системи

Проаналізувавши предметну область та існуючі рішення, можна скласти наступні нефункціональні вимоги до програмного забезпечення:

— розширюваність - наявність багатьох абстракцій та велика функціональність, що дасть змогу додавання нових модулів;

— масштабованість – мікросервісна архітектура, що дозволяє системі працювати при великих навантаженнях, також такий підхід збільшує відмовостійкість, тому що є можливість створення додаткових реплік сервісів. Також в подальшій підтримці та розширенні функціоналу перевагою є спілкування між сервісами, що перевіряє життєвий цикл програмного застосунку;

— переносимість – система має підтримувати перенесення та розгортання її на різноманітних платформах та серверах;

— автоматизоване тестування - під час створення змін проводиться перевірка до готовності програми;

— підтримка багатьма платформами та сучасними браузерами – системи повинна працювати для IOS, Android, Windows, Mac OS, Linux;

— безпека – система має використовувати найсучасніші технології забезпечення безпеки.

3.3 Діаграма використання застосунку

Діаграма використання ІА61.300БАК.005 Д1 була створена для відображення взаємодії користувача з системою. Вона описує можливу функціональність всіх користувачів системи.

За межами прямокутника зображено всіх користувачів автоматизованої системи шкільного процесу у школі:

— адміністратор;

— вчитель;

— учень;

— батько.

В кожного з них є унікальні можливості, які зображено в середині прямокутника.

На схемі можна побачити що основні права користування та заповнення даних (створення, видалення та редагування семестрів, навчального плану, користувачів, звернень) надаються адміністратору та вчителеві.

Батьки та учні в основному мають права для читання, а саме успішності та звернень. Також на схемі можна побачити що відрізняється вхід в систему між адміністраторем на іншими користувачами, тому що вони використовують сторонній додаток Telegram для авторизації. Також варто відмітити що реєстрації немає в даному застосунку. Усі маніпуляції над користувачами проводяться з панелі адміністратора.

3.4 Висновки до розділу

Зважаючи на базу інформації, що була зібрана з попередніх розділів розроблено основні функціональні та нефункціональні вимоги до автоматизованої системи шкільного освітнього процесу, а також створено діаграму використань, на якій описано спроектовані можливості користувачів системи. Варто зазначити що коректно складені вимоги пришвидшують процес розробки.

4 ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ

4.1 Переваги обраних технологій

Розробка та вибір технологій є складним та важливим процесом. Слід дивитися на сучасні тенденції в світі програмування, на взаємодію технологій та відштовхуватися від потрібної функціональності.

Зазвичай розробка програмного забезпечення потребує кількох технологій. Наприклад для створення клієнтської частини часто використовують HTML розмітку та мову програмування JavaScript. Для серверної частини обирають: Scala, Java, C#, Python, тощо. Доволі часто проекти зважаючи на свій розмір діляться на декілька мікро-сервісів, кожен з яких відповідає за певну частину. Усі вони підтримують різноманітні технології.

Вибір залежить від багатьох факторів:

— предметна область;

— нагруженість системи;

— стек технологій команди;

— взаємодія технологій;

— наявність документації;

— вимоги до системи.

При розробці автоматизованої системи шкільного процесу обрано наступні технології:

— мова програмування для серверної частини – Scala;

— мова програмування для клієнтської частини – JavaScript;

— фреймворк Akka;

— СУБД – PostgreSQL;

— бібліотека для взаємодії з Telegram API – Bot4s;

— бібліотека для взаємодії між клієнтською на серверною частинами – jQuery;

— сервіс зберігання проектів – GitHub;

— сервіс для забезпечення неперервної інтеграції – CircleCI.

4.1.1 Мова програмування «Scala»

Scala – мультипарадигмова мова програмування, що поєднує властивості об’єктно-орієнтованого та функціонального програмування. Вихідний код Scala призначений для компіляції в байт-код Java, так що отриманий виконуваний код працює на віртуальній машині Java [1-8].

Особливості Scala:

— використовується для написання широко масштабованих та розподілених систем з великою кількістю обчислень та високою відмовостійкістю;

— синтаксис цієї мови компактний і зазвичай зрозумілий. Це надає змогу розробнику писати код швидко;

— scala компілюється на JVM, що дає змогу уникнути взаємодію з системною архітектурою під час написання;

— взаємодія з кодом Java, також можна використовувати її бібліотеки;

— підтримка «lazy» обчислень, що надає змогу економити ресурси під час роботи програми;

— будь який метод або функція Scala є змінними;

— scala декларативна мова з статичною типізацією що гарантує коректність та відмовостійкість роботи програмного забезпечення.

4.1.2 Фреймворк «Akka»

Akka – це фреймворк розроблений для мов які використовують JVM. Також це інструмент з відкритим вихідним кодом який спрощує написання паралельне і розподілене програмне забезпечення [9-10].

Особливості Akka фреймворку:

— надає готове середовище для будування високонавантажених систем, які ефективно використовують ресурси;

— вбудована можливість кластеризації мікро-сервісів, що допомагає горизонтально розширювати програмне забезпечення;

— зазвичай логіка роботи будується навколо «акторів», в кожного є інформація яку він зберігає та певна функціональність. Всі актори будують велику ієрархію взаємозв’язків;

— є вбудована можливість спілкування між мікро-сервісами за допомогою вузлів кластера по TCP протоколі;

— описана логіка потокової обробки даних, що значно економить час написання вихідного коду;

— будування реактивних відмовостійких систем;

— взаємодія з сторонньою розподіленою системою – Apache Kafka.

4.1.3 Бібліотека «Bot4S»

Bot4s – проста, розширювана, типізована бібліотека, що надає взаємодію з Telegram API[11].

Особливості Bot4s:

— можливість розробки ігор;

— оплата в Telegram;

— взаємодія з Akka;

— реалізація за допомогою implicits;

— готові реалізації контекстів по яких проходить передача даних;

— можливість проведення опитувань;

— можливість використання «webhooks».

В проекті використовується для взаємодії з користувачами, через те що Telegram є кросплатформним мій додаток можна використовувати на багатьох операційний системах та різного роду девайсах. Вся робота телеграм бота у автоматизованій системі освітнього процесу у школі базується на відповідях на різноманітні повідомлення користувачів (вчителів, учнів, батьків), також усі повідомлення мають різну кількість аргументів. Також автоматичне надсилання повідомлень в певні проміжки часу – нотифікації.

4.1.4 СУБД «PostgreSQL»

PostgreSQL– об’єктно-реляційна СУБД. Реалізована для великої кількості платформ: Unix подібних, AIX, BSD, Linux та багато інших [13-14].

Особливості PostreSQL:

— має гнучкий набір налаштувань;

— Multiversion Concurrency Control – підтримка одночасної модифікації бази даних декількома користувачами;

— можливість наслідування таблиць;

— не потребує встановлення додаткових залежностей;

— можливість створення тригерів які запускаються під час виконання певних операцій в алфавітному порядку;

— підтримка наступних індексів: B-дерево, хеш, GiST, GIN, BRIN, Bloom.

Дану систему обрано посилаючись на «benchmarks», що показали PostgreSQL однією з найшвидших реляційних баз даних. Також варто відмітити що бібліотека «Slick», яку використано в даному проекті для взаємодії з базою даних, має вбудовані модулі для роботи з PostgreSQL. В даному проекті розглядалися й NoSQL бази даних, але через структурованість даних було обрано реляційну СУБД – PostgreSQL.

4.1.5 Мова програмування «JavaScript»

JavaScript – мультипарадигмова мова програмування, що підтримує об’єктно-орієнтований, функціональний та імперативний стилі програмування. Широко застосовується в браузерах для надання сторінкам інтерактивності [14-18].

Особливості JavaScript:

— висока швидкість виконання;

— легкий для вивчення та використання;

— автоматичне приведення типів даних;

— функції виступають об’єктами базового класу;

— активна підтримка, що сприяє широкому функціоналу та постійним оновленням;

— велика кількість ресурсів для навчання;

— тестування у браузері;

У даному проекті JavaScript використовується для написання адміністративного інтерфейсу керування, а саме додаткових анімацій та функцій для відправлення та опрацювання даних з API автоматизованої системи оствітнього процесу у школі.

4.1.6 Бібліотека «jQuery»

jQuery – це швидка, невелика та багатофункціональна бібліотека JavaScript. Вона застосовується для зменшення коду і спрощення роботи з web елементами [19].

Особливості jQuery:

— надає API для роботи з AJAX;

— надає доступ до будь якого DOM елемента, звертатися до його атрибутів та та їхнього вмісту, проводити маніпуляції над ними;

— створення візуальних ефектів;

— надає додаткові JavaScript модулі;

— підтримка старих версій браузера;

— надає інструменти для створення різноманітних фільтрів;

— бібліотека з відкритим вихідним кодом, що дає змогу подивитися реалізацію певних функцій та оптимізувати під власні потреби;

— набір інструментів для створення засобів на будь яких пристроях;

— синтаксис для обробки подій.

В даному застосунку використовується для взаємодії певного роду даними між адміністративною панеллю та сервером, для підтримки динамічного обороту даних та візуалізації.

4.1.7 Система контролю версій «GitHub»

GitHub – веб сервіс для хостингу IT-проектів і їхньої розробки, що використовує систему контролю версій Git.

Особливості GitHub:

— надає інтерфейс для поєднання репозиторіїв у вигляді дерева;

— створення репозиторіїв різного рівня доступу;

— можливість додавання файлів за допомогою інтерфейсу;

— підтримує отримання та редагування коду через інші системи контролю версій: Mercurial та SVN;

— зберігання документації у різних форматах файлів типу Markdown;

— показує вклад кожного розробника у вигляді дерева;

— для проектів є особисті сторінки та система відстеження помилок, на сайті присутнє підсвічування синтаксису.

У даному проекті GitHub використано для збереження коду проєкту на приватному репозиторії, а також через те що є інтеграція з Circle CI, що дозволяє проводити автоматизоване тестування та швидко розгортати програмне забезпечення на серверах.

4.1.8 Сервіс забезпечення неперервної інтеграції «CircleCI»

CircleCI – це сервіс для забезпечення неперервної інтеграції. Він використовується для складання проєкту, виконання тестів, розгортання готового проєкту на серверах, хмарних платформах та звітування по виконаній роботі[20].

Особливості CircleCI:

— гнучкі налаштування для ефективного запуску складних; розподілених проектів з можливістю кешування шарів;

— висока оптимізація швидкості складання проєкту;

— має взаємодію з GitHub або Bitbucket;

— постійна підтримка;

— платна версія, що значно збільшує набір інструментів;

— кожне завдання запускається в новому контейнері або віртуальній машині;

— інтеграція сповіщень за допомогою Slack та IRC;

— налаштування для розгортки додатків в різних середовищах: AWS CodeDeploy, AWS EC2 Container Service, AWS S3, Google Kubernetes Engine, Microsoft Azure та Heroku.

CircleCI використовувався при розробці системи для неперервної інтеграції, що дозволяє проводити тестування на кожному етапі, та розгортати проєкт на безкоштовних серверах.

4.2 Висновки до розділу

Зважаючи на предметну область, огляд існуючих рішень та тенденції інструментів для розробки програмного забезпечення були визначені технології для використання під час розробки системи автоматизації шкільного освітнього процесу. Обрані технології добре взаємодіють, що гарантує відмовостійкість та високу швидкість написання проєкту.

5 РОЗРОБКА ЗАСТОСУНКУ

5.1 Структура застосунку

Після завершення проектування та визначення технологій для розробки автоматизованої системи відбувається етап розробки

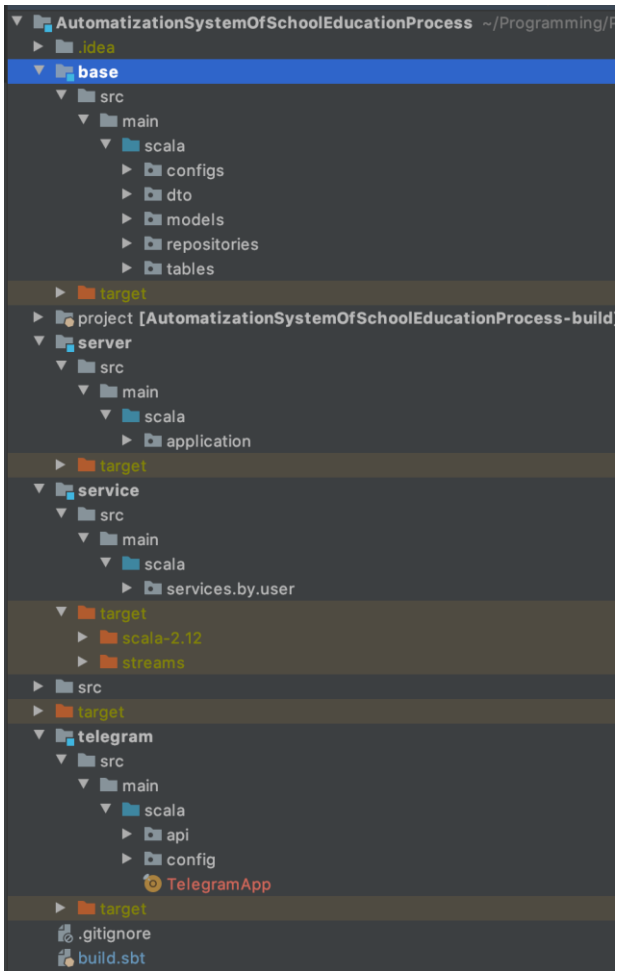

Рисунок 5.1 – Структура застосунку

Структура застосунку яка зображена на рисунку 5.1, складається з наступних компонентів:

— models – опис основних сутностей які використовуються для запису в базу даних та зберігають в собі тотожну інформацію про кожну з моделей;

— dto – опис сутностей які використовуються для передачі між серверною і клієнтською частинами;

— tables – створення відповідностей сутностей з табличками бази даних[17] для налагодження взаємодії;

— repositories – створення відповідних SQL запитів на основі tables до бази даних[18];

— services – основна логічна частина застосунку, проведення певних обчислень, валідацій вхідних даних;

— application – API для взаємодії з клієнтською частиною;

— telegram – створення бота для надання користувачеві можливість зручного інтерфейсу та багатоплатформності.

5.2.1 Діаграма компонентів

Діаграма компонентів ІА61.300БАК.005 Д2 – відображає архітектуру даного застосунку. Клієнтами в даній системі виступають браузер користувача – для адміністратора, а також Telegram для всіх інших користувачів. Серверна частина працює за допомогою веб фреймворка Akka HTTP та проксі сервера Nginx.

5.2.2 Діаграма послідовності

Діаграма послідовності ІА61.300БАК.005 Д3 відображає принцип роботи системи, що наглядно показує виконувані дії при вході адміністратора у систему:

— спочатку користувач в браузері виконує певні дії;

— браузер надсилає запит з даними користувача на необхідну адресу;

— проксі-сервер перенаправляє даний запит на веб сервер безпосередньо застосунку;

— web-застосунок зважаючи на запити користувача проводить певні обчислення;

— після цього web-застосунок генерує відповідь з даними для головної сторінки або повідомленням про помилку, якщо така виникла, та перенаправляє на проксі-сервер Nginx;

— проксі сервер надсилає згенеровану відповідь до браузера;

— браузер відображає отриману сторінку користувачу.

Щодо звичайних користувачів, тут взаємодія з нашим застосунком проводиться за допомогою Telegram, взаємодія та передача даних між телеграмом проходить по захищеному HTTPS протоколу.

5.3 Розробки бази даних

Під час розробки бази даних використано бібліотеку slick для створення запитів та надання доступу. Особливістю slick є вбудована DSL, широке представлення моделей, можливість стрімінгу за допомогою «Reactive Streams». Слід зазначити що Slick містить в собі можливість автоматичного створення табличок, що використана в даному проекті. Також на етапі проектування спроектовано діаграму структури бази даних ІА61.300БАК.005 Д4 наведено, а опис і призначення таблиць бази даних зазначено у таблиці 5.1.

Таблиця 5.1 – Призначення таблиць бази даних

Назва таблиці	Опис
classes	Таблиця зберігає дані про шкільні класи, такі як: ідентифікатор (id), скорочену назву (name), та ідентифікатор на вчителя який закріплений за даним класом.
semesters	Таблиця яка містить в собі записи про навчальний семестр: ідентифікатор (id),

Продовження таблиці 5.1

	дата початку (date_from), дата кінця (date_to) та скорочена назва (name)
users	Містить всі дані про користувачів системи, а саме: ідентифікатор (id), ім’я користувача (name), прізвище (surname), необов’язкове поле по батькові (middle_name), електронна пошта (email), роль, що надає різні права доступу (role), хеш паролю (pass_hash), опціональний ідентифікатор класу з яким користувач взаємодіє (class_id), код для підтвердження пошти (invitation_code), ідентифікатор телеграма для авторизації звичайних користувачів системи, знайти його можна за допомогою спеціального телеграм бота (telegram_id)
subjects	Таблиця відображає шкільні предмети, події, містить дані: ідентифікатор (id), назва предмету (name)
lessons	Містить інформацію про шкільний урок: ідентифікато (id), тема уроку (topic), ідентифікатор предмету (subject_id), ідентифікатор класу (class_id) та ідентиікатор вчителя який проводить подію (teacher_id)

Продовження таблиці 5.1

messages	Таблиця яка містить в собі повідомлення та звернення до певних користувачів, має такі поля: ідентифікатор (id), текст повідомлення (message), дата прибуття (date), ідентифікатор користувача (user_id)
home_works	Містить в собі дані для учнів, а саме: ідентифікатор (id), задача для підготовки до наступного уроку (task), ідентифікатор уроку (lesson_id)
parents_and_pupils	Відтворює взаємодію між батьками і учнями, містить лише ідентифікатор батька (parent_id) та учня (pupil_id)
teachers_and_subjects	Відтворює взаємодію між вчителем та предметом, тобто його спеціалізацію, зберігає дані: ідентифікатор вчителя (teacher_id) та предмету (subject_id)
dailies	Дана табличка зберігає дані про урок як процес, тобто відрізок часу: ідентифікатор процесу (id), час проведення уроку (date), ідентифікатор семестру (semester_id), ідентифікатор предмету (lesson_id)
visits	Дана табличка зберігає дані про успішність учня для кожного уроку, а

Продовження таблиці 5.1

visits	саме: ідентифікатор (id), відвідування (visiting), опціональну оцінку (mark) та ідентифікатор учня (pupil_id)

5.4 Використані шаблони та підходи проектування

Під час розробки системи автоматизації освітнього процесу у школі було використано мультипарадигмову мову програмування Scala, а саме функціональний підхід розробки програмного забезпечення, а також наступні шаблони проектування:

— singleton;

— cake pattern;

— stackable trait;

— last initialization;

— value object.

Варто відмітити що у Scala більшість шаблонів проектування імплементовані за замовчуванням, що дає змогу писати швидкий та лаконічний вихідний код, до таких з вище сказаних можна віднести: singleton, last initialization, value object. Також використовуючи певні бібліотеки, такі як Cats в даному застосунку покращено дизайн за допомогою Design patterns.

5.4.1 Функціональний підхід розробки програмного забезпечення

Функціональний підхід розробки програмного забезпечення - це стиль програмування, який підкреслює написання додатків, використовуючи лише чисті функції та незмінні значення Це декларативний тип стилю програмування. Його головна увага приділяється «що вирішити» на відміну від імперативного стилю, де основний акцент - «як вирішити». Він використовує вирази замість висловлювань. Вираз оцінюється для отримання значення, тоді як оператор виконується для призначення змінних.

Ці функції мають деякі особливості, такі як:

— може приймати функції у вигляді аргументів

— тип функції залежить від останнього значення

— функції мають можливість перенесення (carrying)

— можливість бути анонімною

Також варто відмітити що у них немає побічних ефектів, тобто вони не змінюють будь-які аргументи, глобальні змінні.

5.4.2 Шаблон проектування «Singleton»

«Singleton» – обмежує процес створення класу одним об’єктом і забезпечує глобальну точку доступу до нього. Singleton – найвідоміший шаблон дизайну на Java. Це чітка ознака відсутності функції мови. Scala забезпечує реалізацію Singleton на прикладі об’єкта. Об'єкти можуть успадковувати методи з класів або інтерфейсів. На об'єкт можна посилатися безпосередньо або через успадкований інтерфейс.

В Scala об’єкти для спеціальних класів «case class» створюються автоматично та називаються «об’єкт компаньйон». В цьому об’єкті є метод для створення екземпляру класу.

Особливість case class’ів:

— створений об’єкт компаньйон

— доступ до полів за допомогою pattern matching [1-8];

— відсутність наслідування між ними;

— спеціальне кодування

— класи є рівними у яких ідентичний набір даних

5.4.3 Шаблон проектування «Cake pattern»

«Cake pattern» – DI шаблон проетування, що забезпечує структуровану систему, та можливість швидко знайти потрібні елементи. Шаблон введення

залежності дозволяє уникнути важко кодованих залежностей та замінити залежності або під час виконання, або під час компіляції.

Введення залежностей використовується для вибору серед декількох реалізацій певного компонента в додатку або для забезпечення макетних реалізацій компонентів для тестування одиниць.

У Scala ми використовуємо анотації власного типу разом із ознаками для впровадження введення залежності. На відміну від конструкторського підходу DI, цей підхід вимагає єдиного посилання для кожного виду залежності в конфігурації.

5.4.4 Шаблон проектування «Stackable trait»

Один із способів використання trait у Scala - це модифікація за допомогою «Stackable trait» шаблону проектування. Trait або клас може грати одну з трьох ролей: основу, серцевину або склад. Базовий trait або абстрактний клас визначає абстрактний інтерфейс, на який розширюються всі ядра та стеки (збірник traits які наслідуються між собою).

Основні trait'и або класи реалізують абстрактні методи, визначені в базовому trait'і (інтерфейсі) і забезпечують основну, основну функціональність. Кожна складна частина перекриває один або декілька абстрактних методів, визначених у базовому trait'і, використовуючи абстрактні модифікатори переосмислення Scala та забезпечує деяку поведінку і в якийсь момент викликає «реалізацію батька» того ж методу. Таким чином, склади змінюють поведінку будь-якого ядра, до якого вони змішані.

В основному цей шаблон використовується для послідовної ініціалізації або об'єднання даних по ієрархії наслідувань.

5.4.5 Шаблон проектування «Lazy initialization»

«Lazy initialization» – особливий випадок стратегії лінивої оцінки. Це техніка, яка ініціалізує значення або об'єкт при першому доступі.

Lazy initialization дозволяє відкласти або уникнути дорогого обчислення. У багатопотоковому коді доступ до прапора повинен бути синхронізований для захисту від перегонів. Ефективна синхронізація може використовувати подвійну перевірку блокування, що ще більше ускладнює код. Scala пропонує чистий вбудований синтаксис для визначення лінивих значень. Ліниві значення Scala можуть містити нульові значення. Доступ до лінивих змінних безпечний для потоків.

5.4.6 Шаблон проектування «Value object»

«Value object» – це невеликий незмінний об'єкт, який представляє просту сутність, рівність якої не заснована на ідентичності. Об'єкти значення є рівними, якщо всі їхні поля рівні.

Об'єкти цінності широко використовуються для представлення чисел, дат, кольорів тощо. У корпоративних додатках вони використовуються як DTO для міжпроцесорної комунікації. Через незмінність об'єкти зручні в багатопотоковому програмуванні.

У Scala ми можемо використовувати або кортежі, або класи регістрів, щоб оголосити value objects. Коли окремий клас не потрібен, кортежі - це шлях. Кортежі – це заздалегідь визначені незмінні "колекції", які вміщують фіксовану кількість предметів різного типу. Кортежі забезпечують конструктор та всі допоміжні методи.

5.5 Висновки до розділу

В даному розділі було описано структуру системи з програмної сторони, наведено приклад структури застосунку, та опрацьовано кожен програмний модуль, а також проведено детальний розгляд розроблених таблиць бази даних. Надано короткий опис взаємодії між користувачем та сервером, описано функціональність та зв’язки компонентів системи. Розроблено три схеми: діаграму компонентів, діаграму послідовностей та діаграму структури бази даних.

6 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування програмного забезпечення є важливим етапом життєвого циклу розробки програмного забезпечення. Даний етап потребує детального опрацювання існуючої функціональності розробки. Варто відмітити що тестування є трудомістким процесом, адже він потребує розгляду всіх можливих сценаріїв роботи системи. Існує багато видів тестування. Кожен з видів відрізняється багатьма критеріями, умовно можна виділити наступні питання при виборі виду тестування:

— в залежності від цілей тестування: функціональне тестування (functional), нефункціональне тестування (non-functional), тестування пов’язане зі змінами;

— за знанням системи: тестування чорного ящика (black box), тестування білого ящика (white box), тестування сірого ящика (gray box);

— за ступенем автоматизації: ручне тестування (manual testing), автоматизоване тестування (automated testing), напівавтоматизоване тестування (semiautomated testing);

— за ступенем ізольованості компонентів: модульне тестування (unit testing), інтеграційне тестування (integration testing), системне тестування (end-to-end testing);

— за часом проведення тестування: альфа-тестування (alpha testing), тестування нової функціональності (new feature testing);

— регресійне тестування (regression testing), тестування при здачі (acceptance testing), бета-тестування (beta testing);

— за ознакою позитивності сценаріїв: позитивне тестування (positive testing), негативне тестування (negative testing);

— за ступенем підготовленості до тестування: тестування за документацією (formal testing), інтуїтивне тестування (ad hoc testing).

Для розробки даного застосунку виділено наступні основні види тестування: ручне тестування та автоматизоване. З підвидів автоматизованого обрано модульне тестування та функціональне тестування.

6.1 Ручне тестування

Даний метод тестування полягає в відображенні дій користувача. Тобто основна задача автоматизованих тестів – переконатися що функціональність програмного забезпечення працює правильно. Зазвичай він використовується під час початку написання програмного забезпечення, або добавлення нової функціональності, адже додавання автоматизованого тестування це витратний по часу процес. Також варто відмітити що ручне тестування, не покриває повною мірою весь проєкт. Доволі часто великі компанії випускають бета версії продуктів. Це виконується заради заощадження власних коштів на ручному тестування, а також для задоволення більш широкого спектру можливостей проєкту. Тобто є спеціальні користувачі які хочуть використовувати нову функціональність сайту, що є ризиково.

Ручне тестування під час розробки автоматизованої системи освітнього процесу у школі використовувалося в основному для швидкого виявлення помилок під час написання. Використовувалися популярні методи виявлення помилок: метод інформування проміжних результатів обчислень (logging), а також метод виконання програми з зупинками на певних етапах (breakpoints), що дозволив більш детально проаналізувати шлях (pipeline) вхідних даних до їхнього заключного етапу.

6.2 Автоматизоване тестування

Автоматизоване тестування – процес написання автоматичних тестів, які запускаються перед запуском та розгортанням системи на сервері за допомогою сервісів які надають можливість неперервної розробки.

В даній розробці як сервісом неперервної розробки обрано CircleCI. Також для автоматичного тестування використовується unit testing та functional testing.

Unit testing (модульне тестування) слугує для тестування безпосередньої функціональності даного додатку, без додаткових взаємодій з іншими сервісами.

Тобто результат взаємодії з іншими сервісами описується як існуючий, а етап обчислення як очікуваний.

Functional testing (функціональне тестування) використовується для тестування повної взаємодії сервісів. В даному проекті для взаємодії з Telegram та базою даних. Тобто відтворено основні етапи спілкування звичайних користувачів та сервісів застосунку.

Для спрощення написання автоматизованих тестів розробляється структура (план), в якій описують можливі тестові випадки (test case) програмного забезпечення.

У таблицях 6.1 – 6.17 наведено основі тестові випадки.

Таблиця 6.1 – Тестовий випадок «Перевірка відображення навчального плану»

Дія	Дізнатися навчальний план на даний момент часу
Очікуваний результат	— сторінка з навчальним планом на 2 тижні відкрита; — відображено назву теперішнього семестру — відображено клас для якого зроблено запит
Результат тесту	Успішний

Таблиця 6.2 – Тестовий випадок «Перевірка створення користувача»

Дія	Створити нового користувача, на сторінці адміністратора
Очікуваний результат	— відкрита сторінка створення користувача — відображено список всіх користувачів в якій з'явився новий — в користувача на пошті з’явився лист — в листі присутня кнопка підтвердження
Результат тесту	Успішний

Таблиця 6.3 – Тестовий випадок «Перевірка створення предмету»

Дія	Створити нового предмету, на сторінці адміністратора
Очікуваний результат	— відкрита сторінка створення предмету — відображено список всіх предметів в якій з'явився новий
Результат тесту	Успішний

Таблиця 6.4 – Тестовий випадок «Перевірка створення класу»

Дія	Створити нового предмету, на сторінці адміністратора
Очікуваний результат	— відкрита сторінка створення класу — відображено список всіх класів в якій з'явився новий
Результат тесту	Успішний

Таблиця 6.5 – Тестовий випадок «Перевірка призначення предмету певному вчителю»

Дія	Розширити спеціалізацію викладача або вчителя
Очікуваний результат	— відкрита розширення спеціалізації — повідомлення про успішно призначення нової спеціалізації — відображено список всіх предметів які входять в стек вчителя
Результат тесту	Успішний

Таблиця 6.6 – Тестовий випадок «Імпортування навчального плану»

Дія	Імпортувати навчальний план з файла
Очікуваний результат	— відкрита сторінка створення навчального плану — в випадаючому списку показано частковий навчальний план — повідомлення про успішне імпортування даних
Результат тесту	Успішний

Таблиця 6.7 – Тестовий випадок «Створення звернення або повідомлення до певного користувача»

Дія	Створити звернення
Очікуваний результат	— відкрита сторінка створення звернення для користувачів — отримувач отримав повідомлення в — зазначений час
Результат тесту	Успішний

Таблиця 6.8 – Тестовий випадок «Зміна навчального плану для певного дня»

Дія	Зміна навчального плану
Очікуваний результат	— відкрита сторінка зміни навчального плану — повідомлення про успішну зміну навчального плану
Результат тесту	Успішний

Таблиця 6.9 – Тестовий випадок «Створення домашнього завдання за допомогою Telegram»

Дія	Створення домашнього завдання вчителем
Очікуваний результат	— повідомлення про успішне створення домашнього завдання
Результат тесту	Успішний

Таблиця 6.10 – Тестовий випадок «Створення успішності учня за допомогою Telegram»

Дія	Створення успішності учня
Очікуваний результат	— повідомлення про успішне створення успішності учня
Результат тесту	Успішний

Таблиця 6.11 – Тестовий випадок «Відображення успішності для учня або батьків в messenger»

Дія	Відображення успішності
Очікуваний результат	— повідомлення про успішність учня за один день; — кнопка для розгляду успішності за довгий період часу; — кнопка для розгялду успішності для певних користувачів.
Результат тесту	Успішний

Таблиця 6.12 – Тестовий випадок «Відображення домашнього завдання для учнів в Telegram»

Дія	Відображення домашнього завдання
Очікуваний результат	— повідомлення в якому міститься домашнє завдання на наступний день — кнопка для розгляду домашнього завдання за інший період часу
Результат тесту	Успішний

Таблиця 6.13 – Тестовий випадок «Експортування успішності за певний період часу»

Дія	Експортування успішності учнів
Очікуваний результат	— сторінка з налаштуваннями для гнучкого експортування даних — завантажений файл містить коректні дані
Результат тесту	Успішний

Таблиця 6.14 – Тестовий випадок «Перевірка відображення сторінки після авторизації адміністратора»

Дія	Авторизація адміністратора
Очікуваний результат	— головна сторінка з меню для вибору наступних кроків — відображення прізвища поточного адміністратора — відображення кнопки logout
Результат тесту	Успішний

Таблиця 6.15 – Тестовий випадок «Перевірка виходу з системи адміністратора»

Дія	Logout адміністратора
Очікуваний результат	— головна сторінка з назвою застосунку — кнопка для авторизації адміністратора — кнопка для скидання паролю
Результат тесту	Успішний

Таблиця 6.16 – Тестовий випадок «Авторизація користувача в Telegram»

Дія	Авторизація користувача
Очікуваний результат	— повідомлення про успішну авторизацію — повідомлення з привітанням користувача — повідомлення про можливості користувача в системі
Результат тесту	Успішний

Таблиця 6.17 – Тестовий випадок «Нагадування про урок»

Дія	Авторизація користувача
Очікуваний результат	— в очікуваний відлік часу отримання нагадування про початок уроку — кнопка для відключення нотифікацій
Результат тесту	Успішний

6.3 Неперервна інтеграція

Неперервна інтеграція – це можливість розробки програмного забезпечення під час якої автоматизоване тестування, розгортання застосунку на різних

платформах, відправлення звітів кожного з етапів відбувається автоматично, за допомогою спеціальних сервісів. Це необхідно для пришвидшення часу розробки, зменшення коштів на персонал який займається цим.

Процес впровадження наступний:

— весь вихідний код потрібно завантажити на репозиторій системи контролю версій, а також додаткові файли з налаштуваннями для тестування, розгортання проєкту на певній платформі;

— підключити сервіс до системи контролю версій та налаштувати під свої вимоги.

В даному проекті використано CircleCI для неперервної інтеграції, та синхронізував її з проектами які зберігаються на GitHub.

6.4 Висновки до розділу

В даному розділі обрано види тестування автоматизації освітнього процесу у школі, а також описано основі тестові випадки. Також розглянуто впровадження автоматизованих тестів під час розробки з використанням неперервної інтеграції.

7 ВПРОВАДЖЕННЯ ТА ВИКОРИСТАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

7.1 Апаратні та програмні вимоги для експлуатації програми

Дана автоматизована система розроблялася мовою Scala. Через те що дану мову програмування використовують зазвичай високонавантажені системи, відповідно системні вимоги є високими, але для даного застосунку це не так.

У таблицях 7.1 та 7.2 наведені мінімальні та рекомендовані вимоги до сервера.

Таблиця 7.1 – Мінімальні вимоги до програмно-апаратного забезпечення системи

Процесор	2-ядерний процесор з тактовою частотою 1.4 ГГц, величиною кеша в 2 мегабайта
Місце на диску	1-2 гігабайти
Оперативна пам’ять	1 гігабайт
Операційна система	Windows 7, macOS та Linux

Таблиця 7.2 – Рекомендовані вимоги до програмно-апаратного забезпечення системи

Процесор	4-ядерний процесор з тактовою частотою 2.8 ГГц, величиною кеша в 4 мегабайта
Місце на диску	5-10 гігабайти
Оперативна пам’ять	4 гігабайт
Операційна система	Windows 10, macOS Catalina та Linux

Через те що Scala працює на Java Virtual Machine (JVM), що потребує наявності java для роботи цього застосунку. Під час написання автоматизованої

системи шкільного процесу використовувалася java 11, тому для коректної роботи потрібно її або версії вище.

Клієнтськими частинами виступаються браузер та додаток Telegram. Через те що клієнтська частина розроблена за допомогою мови програмування JavaScript, вимоги браузера, а саме його версії відповідають вимогам даної мови програмування, що наведено у таблиці 7.3

Таблиця 7.3 – Вимоги до браузера, для коректної роботи клієнтської частини

Назва браузера	Мінімальна версія
Chrome	77 і вище.
Firefox	69 і вище.
Safari	12 і вище
Opera	60 і вище.
Internet Explorer	12 і вище.
Microsoft Edge	76 і вище.
Chromium	43 і вище.
Android Browser	версія андроїд 3.2 і вище.
Amigo	26 і вище.
Tor Browser	10 і вище.
UC Browser	45 і вище.

7.2 Інструкція з встановлення

У зв’язку з тим що дана автоматизована система використовує сервіс який забезпечує непреривну інтеграцію є можливість налаштувати CircleCI для

розгортки застосунку на сервері або платформі. Також якщо не має досвіду з використанням Circle CI є можливість розгорнути систему власноруч за допомогою Docker-compose.

Також є варіант послідовного встановлення необхідних вимог для запуску даного застосунку, а саме Java Runtime Environment (JRE), та виконати команду в терміналі java -jar «назва jar файлу». Також для клієнтської частини необхідно підняти локальний сервер за допомогою Nginx.

7.2 Інструкція для користувача

Користувачів даного застосунку можна умовно поділити на дві групи. Перша група це адміністратори, а друга це звичайні користувачі, а саме учні, вчителі та батьки. Варто відмітити що різниця не тільки в правах доступу, а й в клієнтах які використовують кожен з учасників груп. Для першої групи клієнт розроблений за допомогою JavaScript, що дає змогу використовувати браузер для взаємодії з системою, для другої групи клієнтом виступає додаток Telegram, також можна використовувати його онлайн версію в браузері. Нижче описано детальну взаємодію з системою для кожної з груп.

7.2.1 Інструкція для адміністратора

Після розгортки на сервері даного застосунку у адміністратора є змога перейти на основну сторінку, що зображена у додатках на рисунку Б.1 та авторизуватися під логіном та паролем які встановленні за замовчуванням. Після авторизації рекомендовано змінити дані для входу та збільшити кількість адміністраторів. Варто зазначити що даний застосунок має змогу використовувати захищений HTTPS протокол, який налаштовується за допомогою NGINX, що забезпечує безпечну передачу даних між адміністратором та системою. Після зміни налаштувань за замовчуванням, потрібно підтвердити дані дії за допомогою електронної пошти та одноразового коду.

Варто зазначити що після авторизації для адміністратора відкривається основна сторінка проєкту з навігацією по спроектованій функціональності. Умовно функціональність можна розбити на декілька основних груп, для кожної з них в застосунку є окремі сторінки:

— користувачі;

— класи;

— семестри;

— предмети;

— навчальний план;

— звернення;

— успішність;

— налаштування.

Кожна із наведених вище сторінок містить свій набір інструментів. На сторінці користувачі. (users). адміністратор має можливість додавати нових користувачів в систему. Варто відмітити що адміністратори мають однаковий рівень доступу, тобто не має super-user, який може брати відповідальність за створення нових адміністраторів. Для кожної групи доступу користувачів існує свій набір обов'язкових полів які потрібно заповнити. На рисунку 7.1 зображено вікно взаємодії з користувачами системи, та показано основні поля. Випадаючий список «Select a role», надає змогу обрати рівень доступу користувача.

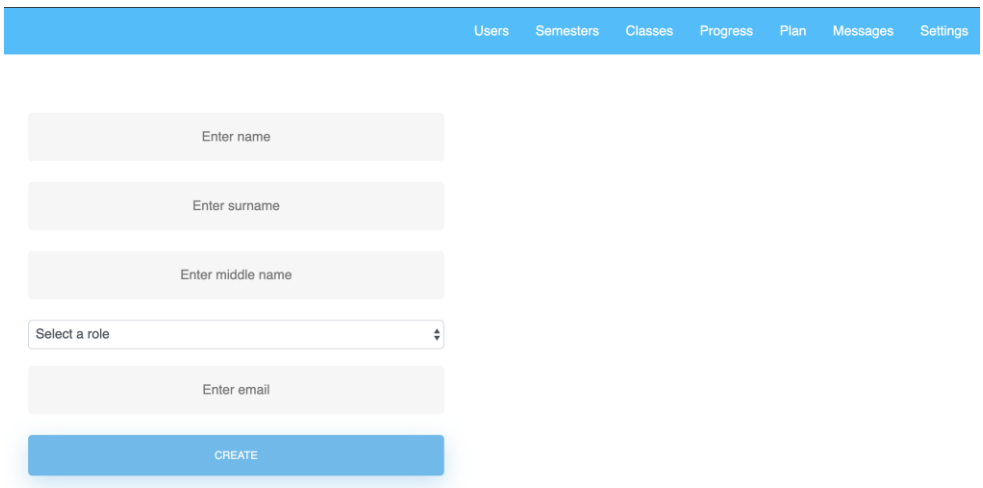

Рисунок 7.1 – Вікно взаємодії з користувачами системи

Також після створення користувача в системі, йому необхідно верифікувати акаунт, за допомогою листа на пошті з спеціально згенерованим кодом або добавити телеграм бота. Також у адміністратора є права видалення та редагування деяких полів існуючих користувачів. Не рекомендується видаляти користувачів які завершили навчання, задля збереження історії. Є можливість зробити їх неактивними. Тобто це певний прапорець який заперечує будь які взаємодії даного користувача з системою.

Наступною вкладкою після користувачів є вкладка семестри, що дає змогу створювати нові навчальні періоди, для яких описується план на певний відлік часу. Вона зображена на рисунку 7.2.

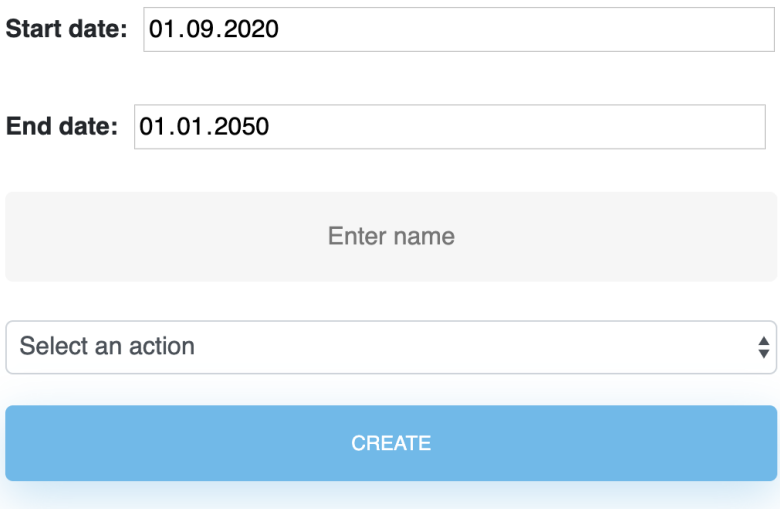

Рисунок 7.2 – Вікно взаємодії з семестрами

Сторінка класи дає змогу об’єднувати учнів в певні групи. Для кожного класу складається раніше визначений навчальний план. Також у адміністратора є можливість редагування класів, що дає змогу розширювати кількість учасників, та змінювати викладача який відповідає за даний клас. Сторінка маніпуляції з класами зображена у додатках на рисунку Б.2.

Сторінка предмети, що зображена у додатках на рисунку Б.3 дозволяє адміністратору проводити маніпуляції над шкільними предметами, а саме створення, видалення та редагування. Варто відмітити що видалення не працює, якщо даний предмет уже використовується та для нього створена успішність.

Також у адміністратора є можливість імпортування структурованих даних, що значно пришвидшить створення, якщо в системі уже зареєстровані предмети з такими назвами, їхнє створення буде проігноровано. Дані імпортуються в форматі tsv, знак розділення полів в даному файлі виступає tab. Також в застосунку є валідатор який повідомляє про некоректність введених даних. Випадаючий список надає можливість адміністратору вибирати дію на існуючими класами, а саме:

— створення;

— видалення;

— редагування.

Сторінка навчального плану працює ідентично до сторінки маніпуляції предметами. Тобто тут також є імпортування даних і валідація, адже в ручному форматі дуже важко заповнити навчальний план для всієї школи. Також є можливість повторення даних на певний період часу, тобто адміністратор заповнив коректні дані для двох тижнів та обрав функцію повторення до закінчення семестру. Система автоматично створила план для кожного дня на основі даних з двох тижнів. Після створення плану у адміністратора є можливість передсвічитися коректністю роботи застосунку та переглянути навчальний план в зручному для себе форматі: в браузері або експортувати дані в файл. Вікно взаємодії з навчальним планом відображено на рисунку 7.3.

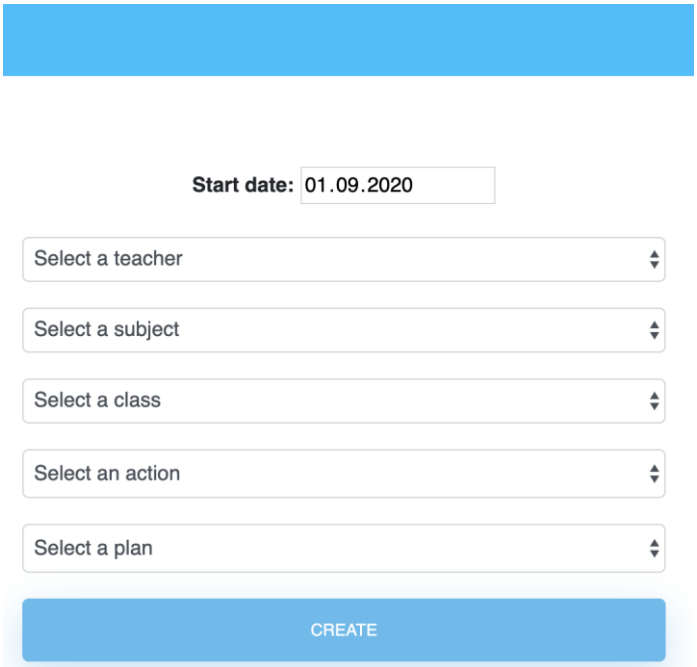

Рисунок 7.3 – Вікно взаємодії з навчальним планом

Сторінка звернення відповідає за створення важливих повідомлень, що інформують інших користувачів. Для створення необхідно зазначити час відправки повідомлення та обрати адресата. Вікно взаємодії з зверненнями зображено на рисунку Б.4. Також варто відмітити що на основі повідомлень будуються нотифікації для учнів, тобто спеціальні повідомлення які інформують учнів про початок уроку або будь якої події. Також є можливість зробити глобального оголошення на основі повідомлень, тоді інформація розподіляється між усіма користувачами системи.

Сторінка успішність надає змогу слідкувати за відвідуваннями та оцінками учнів в системі. Ця інформація є певною мірою публічною, адже вчителі також маються доступ до успішності своїх учнів, кожен учень має можливість дізнатися власну успішність, та батьки можуть слідкувати за даними своїх дітей. Дана сторінка має багато фільтрів, а саме:

— часовий – надає успішність за певний період часу;

— окремих груп – надає успішність окремого класу;

— персональний – успішність для окремої людини;

— предметний – успішність за певним предмету;

— критерії оцінок;

— збірний, що об’єднує попередні фільтри.

Також у адміністратора є можливість експортування даних в файл. Сторінка успішності відображена на рисунку 7.4.

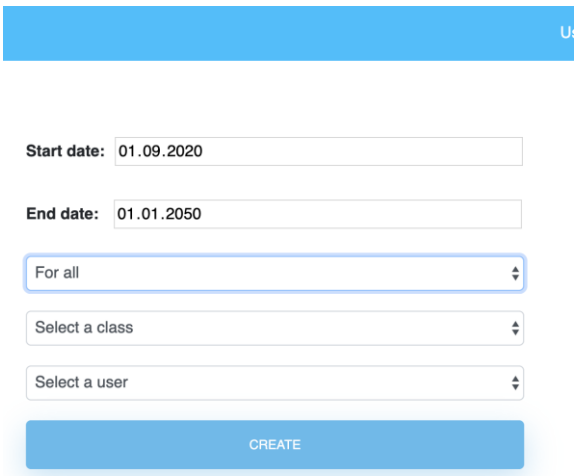

Рисунок 7.4 – Вікно відображення успішності учнів

Остання сторінка – налаштування, дає змогу користувачеві змінити логін та пароль, а також всі інші поля які є публічними та вийти з системи. Вона зображена на рисунку 7.5. Варто відмітити що API підтримує функцію часткового редагування, тобто поля які не були заповнені залишаться в базі даних з такими ж значеннями.

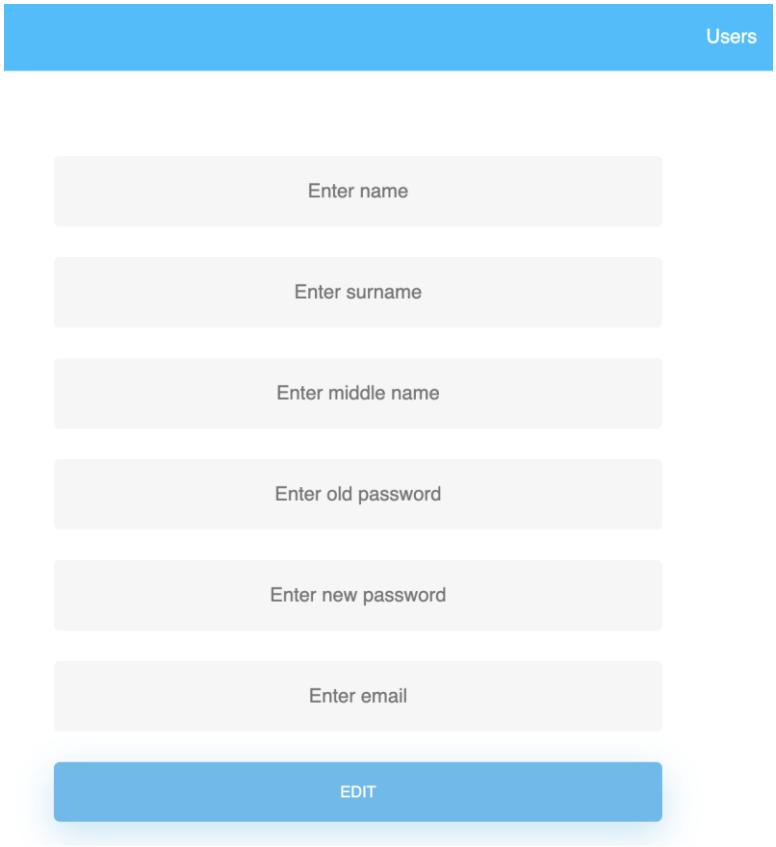

Рисунок 7.5 – Вікно налаштувань

7.2.2 Інструкція для звичайних користувачів

Усі інші користувачі системи використовують Telegram бота для взаємодії з системою автоматизації шкільного освітнього процесу. В кожного з них відрізняється функціональність. Для початку взаємодії з ботом системи, потрібно щоб користувач з його номером телефона уже був добавлений адміністратором. Дальше потрібно ввести команду «/start», після чого бот привітається та

ознайомить користувача з його можливостями в системі. На рисунку 7.10 вікно взаємодії з телеграм ботом та набір функціональності на прикладі учня.

Для учня набір функціональності складає:

— перегляд домашнього завдання;

— перегляд успішності за певний період часу;

— перегляд звернень від адміністратора;

— створення звернень.

Для вчителя набір функціональності складає:

— створення успішності учня;

— перегляд успішності певного учня;

— перегляд успішності певного класу;

— перегляд успішності за певний день;

— створення звернень до учнів та батьків.

Функціональність для батьків:

— перегляд успішності учня який прив’язаний до них в системі;

— перегляд звернень.

Також варто відмітити що у учня та вчителя є можливість отримання нотифікацій, тобто новин про початок уроку або певних подій. Для того щоб змінити аккаунт в системі, необхідно звернутися до адміністратора, та повідомити новий номер, або telegram id. Для початку взадємодії з ботом запускаємо команду «/start», після цього очікуємо відповідь від бота у вигляді привітання. Дальше користувачеві потрібно ввести команду «/help» – що надає список існуючих команд. У кожної команди є своя назва та опис. Щоб дізнатися опис будь-якої команди потрібно скористаться останньою («/description»), що зображена на рисунку 7.6. Після введення команди «description», необхідно ввести назву команди, яка цікавить користувача. Telegram бот автоматично надішле повідомлення з описом потрібної користувачеві команди. Потрібно уважно читати опис команди, тому що деякі з них можуть приймати різну кількість аргументів, які є обов’язковими.

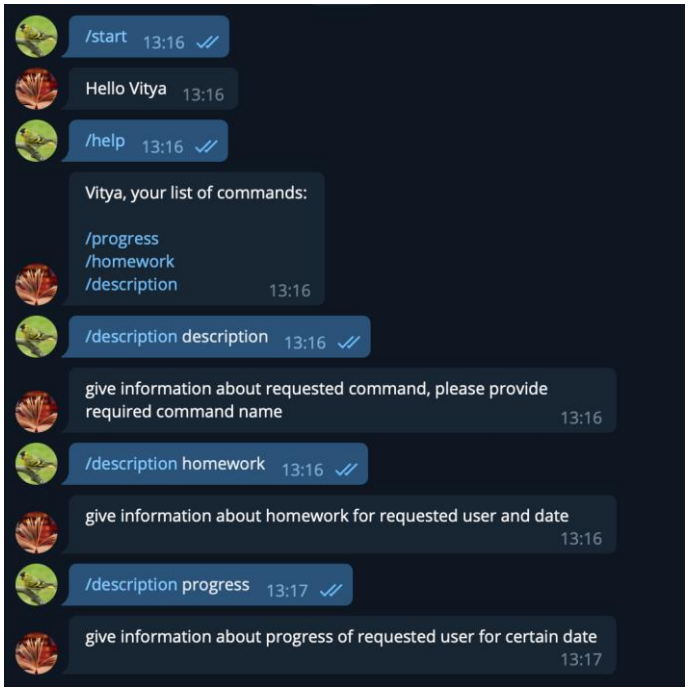

Рисунок 7.6 - Вікно взаємодії з телеграм ботом

7.3 Висновки до розділу

Визначення системних та програмних вимог заключний етап розробки програмного забезпечення. Об’ємна документація з встановлення та документація для користувачів системи є обов'язковою потребою сучасних проектів. Правильно складено інструкція дає змогу користуватися систему не досвідченим особам, а також відповідає на питання стосовно функціональності та інтерфейсу.

ВИСНОВКИ

При виконанні дипломного проєкту було створено автоматизовану систему шкільного освітнього процесу.

Спочатку було проведено аналіз предметної області даної системи. Було визначено призначення, цілі та задачі розробки, а також було визначено основні особливості та критерії користування майбутньої системи. Далі було проведено аналіз існуючих рішень, було виділено переваги та недоліки кожного, на їхній основі створено опис. Зважаючи на отримані дані було розроблено функціональні та нефункціональні вимоги до системи. Під час проектування системи було проведено вибір технологій розробки, з детальним обґрунтуванням цього вибору. На основі вимог до системи, визначених раніше було розроблено автоматизовану систему. Після розробки системи було створено тестові випадки, які тестували не тільки застосунок, а й взаємодію з іншими сервісами. На їх основі проведено тестування. У заключній частині було описано системні та програмні вимоги до даного застосунку, а також написана документація з встановлення та інструкції для користувачів в системі.

Дана автоматизована система готова до застосування та задовольняє всі визначені вимоги. Варто відмітити, що система має переваги: відмовостійкість, високий ступінь безпеки, об’ємну функціональність. До основних недоліків можна віднести: графічний вигляд системи та високі апаратні вимоги. Також є змога розширення функціональності для збільшення автоматизації освітнього процесу.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cay Horstmann. Scala for the impatient. 2017 – 414 с.

2. Paul Chiusano and Runar Bjarnason. Foreword by Martin Odersky. Functional Programing in Scala. 2014 - 320 p..

3. Martin Odersky, Lex Spoon and Bill Venners. Programming in Scala: A Comprehensive Step-by-Step Guide, Third Edition. 2008 – 516 p.

4. Joshua D. Suereth. Foreword by Martin Odersky. Scala in Depth. 2012 – 304 p.

5. Dean Wampler and Alex Pain. Programming Scala: Scalability = Functional Programming + Objects 2nd Edition. December 14 2014 – 586 p.

6. Alvin Alexander. Scala Cookbook: Recipes for Object-Oriented and Functional Programming 1st Edition. 2013 – 722 p.

7. Jameson Schwarz. Learning Scala: Practical Functional Programming for the JVM 1st Edition. 2014 – 256 p.

8. Daniel Hinojosa. Testing in scala 1st edition. 2013 - 166 p.

9. Akka (toolkit) Akka Team. «Akka 2.6.3 Released». Retrieved 4 February 2020

10. Фрейморка Akka (toolkit): [Електронний ресурс] - Режим доступу: https://akka.io/

11. Документація по використанні Bot4s: [Електронний ресурс] – Режим доступу: https://index.scala-lang.org/bot4s/telegram/telegrambot4s/3.0.15?target=_2.12

12. Игорь Дьяков, Алексей Дьяков. Проектирование баз данных: разработка, оптимизация, программирование. 2019 – 128 с.

13. Брешенков Александр. Базы данных: Проектирование баз данных на основе информации табличного вида (Russian Edition). 2011 – 404 с.

14. Justin Scherer. Hands-On JavaScript High Performance: Build faster web apps using Node.js, Svelte.js, and WebAssembly. 2020 – 376 p.

15. Learn JavaScript: [Електронний ресурс] – Режим доступу: https://learn.javascript.ru/

16. Kyle Simpson. You Don't Know JS Yet: Get Started Kindle Edition. 2020 – 145 p.

17. Adam D. Scott. JavaScript Everywhere: Building Cross-Platform Applications with GraphQL, React, React Native, and Electron 1st Edition. 2020 - 344 p.

18. Документація jQuery: [Електронний ресурс] – Режим доступу: https://api.jquery.com/

19. Адам Фримен. jQuery для профессионалов = Pro jQuery. – М.: «Вильямс», 2012. — 960 с

20. CircleCI: [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/144462/ - Назва з екрану.

ДОДАТОК А

Код програми

```
trait RoutePacks {
  self: RouteSimplification
    with RouteWithMapSimplification
    with Directives
    with FailFastCirceSupport =>

  protected def simplePack[Entity](
    pathName: PathMatcher[Unit],
    repository: CrudRepository[Entity],
    routes: Route*
  )(implicit encoder: Encoder[Entity], decoder: Decoder[Entity]): Route =
    path(pathName) {
      postEntity(repository) ~ concat(routes: _*)
    }

  protected def simplePackDTO[Entity <: Id[Int], EntityConverter](
    pathName: PathMatcher[Unit],
    repository: BaseRepository[Int, Entity, _],
    convert: EntityConverter => Entity,
    routes: Route*
  )(implicit encoder: Encoder[Entity], decoder: Decoder[Entity], dtoDecoder:
Decoder[EntityConverter]): Route =
    path(pathName) {
      concat(
        getEntities(repository),
        getEntity(repository),
        postEntity(repository, convert),
        putEntity(repository),
```

```
        deleteEntity(repository),
      ) ~ concat(routes: _*)
    }

  protected def hardPackDTO[Entity <: Id[Int], EntityConverter,
EntityCreateConverter](
    pathName: PathMatcher[Unit],
    repository: BaseRepository[Int, Entity, _],
    fromConverter: EntityConverter => Entity,
    fromCreateConverter: EntityCreateConverter => Entity,
    convert: Entity => EntityConverter,
    routes: Route*
  )(
    implicit encoder: Encoder[EntityConverter],
    decoder: Decoder[EntityConverter],
    dtoCreateDecoder: Decoder[EntityCreateConverter]
  ): Route =
    path(pathName) {
      concat(
        getEntities(repository, convert),
        getEntity(repository, convert),
        postEntity(repository, fromCreateConverter),
        putEntity(repository, fromConverter),
        deleteEntity(repository)
      ) ~ concat(routes: _*)
    }
}

trait RouteSimplification {
  self: Directives with FailFastCirceSupport =>
```

```
  protected def getEntities[Entity](
    repository: CrudRepository[Entity]
  )(implicit encoder: Encoder[Entity]): Route = get {
    onComplete(repository.get()) {
      case Success(entities)  => complete(entities)
      case Failure(exception) =>
complete(HttpResponse(StatusCodes.InternalServerError))
    }
  }

  protected def getEntity[Entity <: Id[Int]](
    repository: BaseRepository[Int, Entity, _]
  )(implicit encoder: Encoder[Entity]): Route = get {
    parameters("id".as[Int]) { id =>
      onComplete(repository.get(id)) {
        case Success(Some(entity)) => complete(entity)
        case Success(_)            => complete(HttpResponse(StatusCodes.NotFound))
        case Failure(exception)    =>
complete(HttpResponse(StatusCodes.InternalServerError))
      }
    }
  }

  protected def postEntity[Entity](
    repository: CrudRepository[Entity]
  )(implicit decoder: Decoder[Entity]): Route = post {
    entity(as[Entity]) { entity =>
      onComplete(repository.create(entity)) {
        case Success(_)         => complete(HttpResponse(StatusCodes.OK))
        case Failure(exception) =>
```

```
complete(HttpResponse(StatusCodes.InternalServerError))
      }
    }
  }

  protected def putEntity[Entity <: Id[Int]](
    repository: BaseRepository[Int, Entity, _]
  )(implicit decoder: Decoder[Entity]): Route = put {
    entity(as[Entity]) { entity =>
      onComplete(repository.update(entity)) {
        case Success(_)         => complete(HttpResponse(StatusCodes.OK))
        case Failure(exception) =>
complete(HttpResponse(StatusCodes.InternalServerError))
      }
    }
  }

  protected def deleteEntity[Entity <: Id[Int]](repository: BaseRepository[Int, Entity,
_]): Route = delete {
    parameters("id".as[Int]) { id =>
      onComplete(repository.remove(id)) {
        case Success(_)         => complete(HttpResponse(StatusCodes.OK))
        case Failure(exception) =>
complete(HttpResponse(StatusCodes.InternalServerError))
      }
    }
  }
}
```

ДОДАТОК Б

Графічний інтерфейс

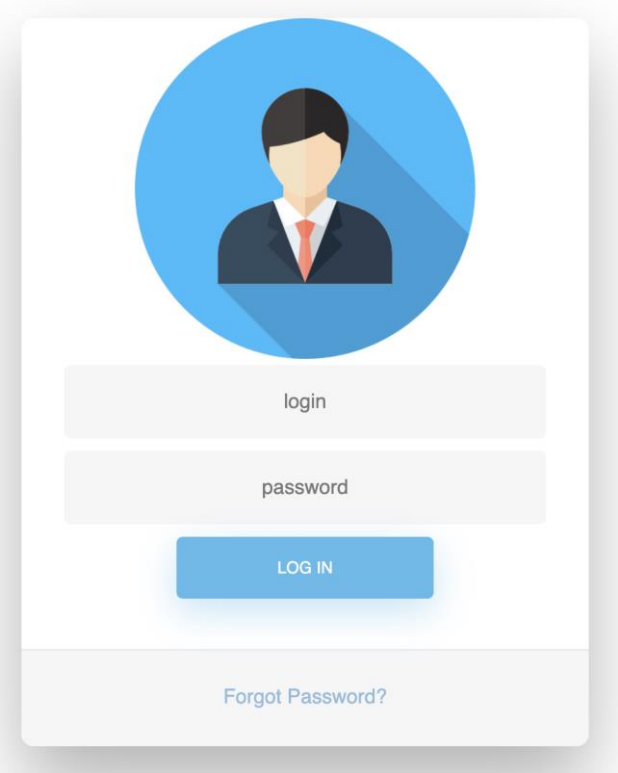

Рисунок Б.1 – Вікно авторизації адміністратора

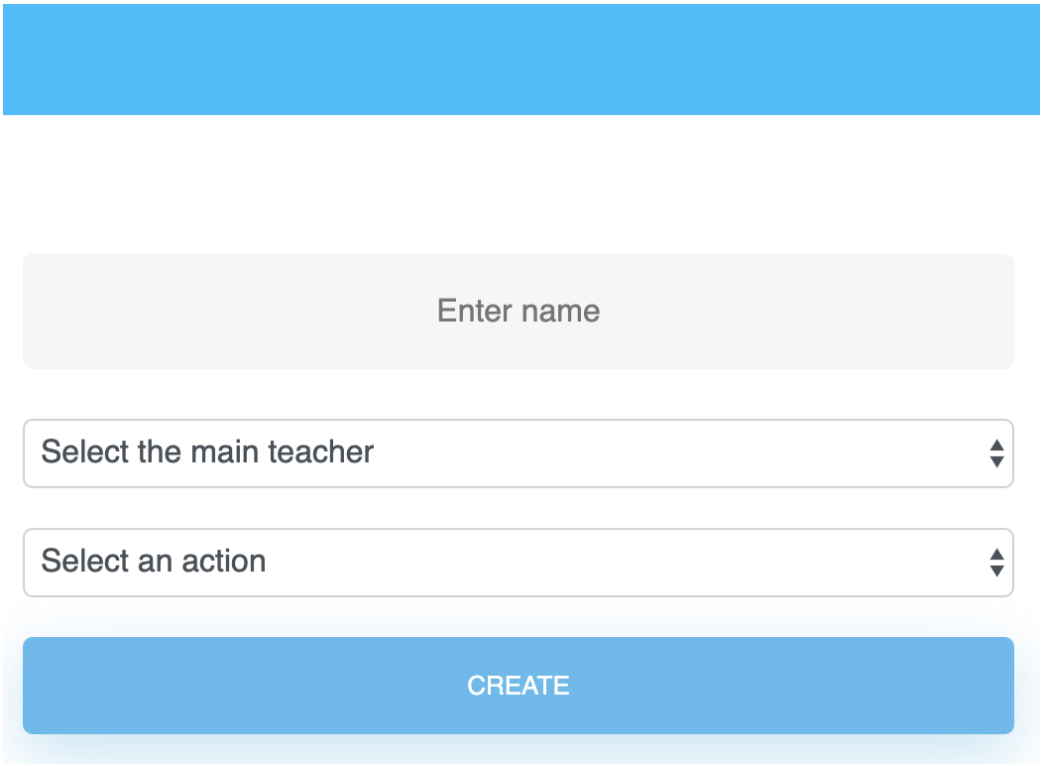

Рисунок Б.2 – Вікно взаємодії з класами

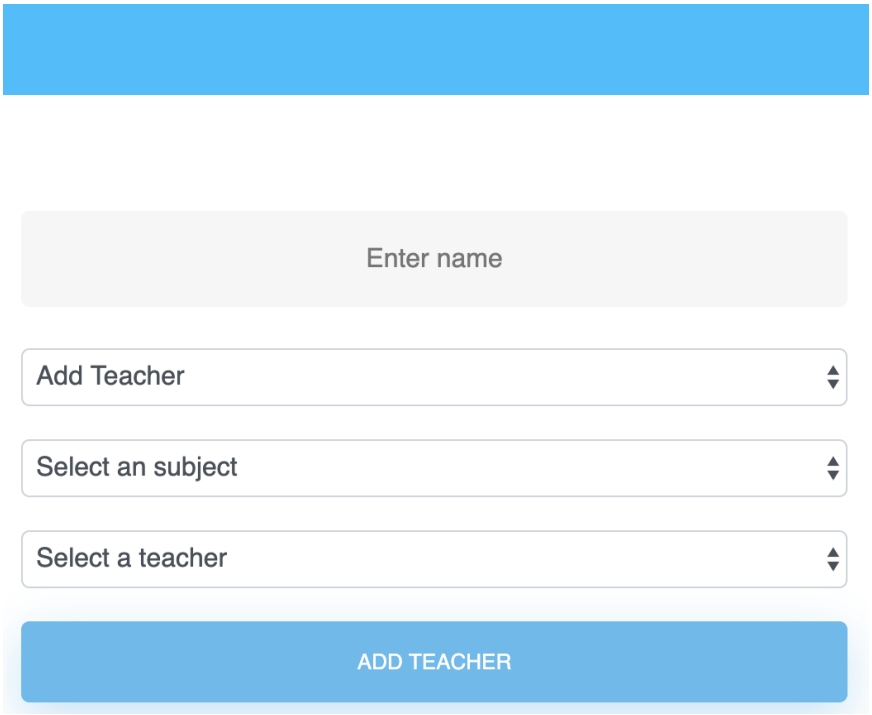

Рисунок Б.3 – Вікно взаємодії з предметами

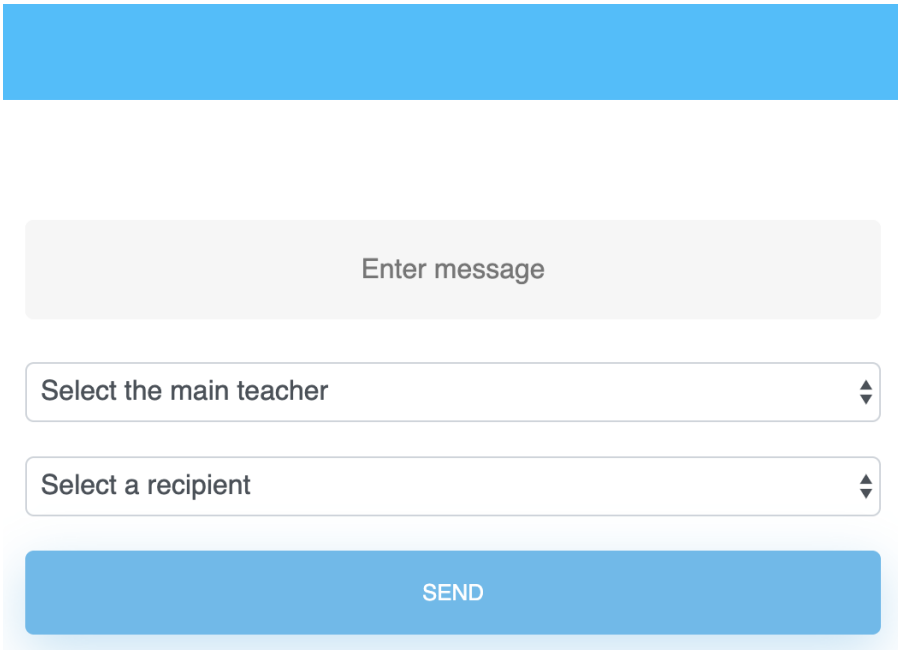

Рисунок Б.4 – Вікно взаємодії з зверненнями